Lab 13: some practical project management tips

Statistics for Social Scientists II

Bur, GJM

2024-11-25

Contents

1 Project management generally

Project management is a central part of the research process, but a sadly underrated one.

In what follows, I'll share some tips based on my own experience—so, I'm not picking on anyone in particular besides myself! Note that many of these were mentioned in my lab syllabus. Hopefully after a lot of gentle encouragement over the semester and lots of chances to practice these, those tips make a bit more sense.

2 Hotkeys/keyboard knowledge

First, learn the hotkeys! That's how I zip around so fast on the computer in lab :). Here are some worthwhile hotkeys to know; these will depend on your

operating system:

- Reload a page (macOS: `Cmd` + `r`).
- Open a program (macOS: `Cmd` + `spacebar`, then search).
- Close a window (macOS: `Cmd` + `w`).
- Reload a page (macOS: `Cmd` + `r`).
- Jump to the URL bar to the URL bar (macOS: `Cmd`+ `l`)
- Tab between programs (macOS: `Cmd`+`Tab`)
- Tab between tabs (macOS: `Cmd` + {number of window; 9 is either ninth *or* the last window that you have open).
- Execute parts of a program (Stata: highlight and `Cmd`+`d`; most others, highlight and `Cmd` + `Enter`).
- Jump to the end of a line (macOS: `Cmd` + `->`) or farther (macOS: `Cmd` + `↓`).
- Select to the end/beginning of a line (macOS: `shift` + `Cmd` + `->`).

3 Develop a coherent file management system.

Here are some general tips for managing your files.

- Make sure you understand what data storage *is* on a fundamental level. Where are your files stored—on your local machine? On a remote server (or "the cloud"), either through a service like Google Drive or through a remote desktop? In general, make sure that you know what is on your machine vs. elsewhere.
- Make folders for different projects. Store information in logical places; don't just pollute your `/downloads` folder.
- Version-control your files (this means clearly keeping track of successive versions). A simple way to do this is to date your files the way I recommend, which allows for easy comparison; the advanced version is to use Git, which I generally recommend. Git not only allows you to easily version-control your files by automatically "versioning" them, it also stores small amounts of data remotely for free. You can probably at least store your generally-very-small do-files and PDFs on a Github.
- In some way, back up your data regularly. An external hard drive is the best method for most things (music, photos, PDFs), but Github works well for small files. Proprietary cloud services such as Google Drive can work well, but make sure that you understand their rules of operation; for example, if you delete a local file and then sync the local machine with the remote machine with the local, does this delete the file off of the remote?

4 Data management

Make sure that all data cleaning is documented and done to the greatest extent possible in a script. Here I will also partially describe what *not* to do, in a sense,

by contrasting it to the workflow an inexperienced user might use (such as a past version of me).

- Pull all of your data in with a script. In other words, *do not* rely on manually opening a data-set with point-and-click, otherwise known as the "graphical user interface" or GUI.[1] You *will* forget exactly which version of the data you used. Stata, R, Python—any modern statistical software can effortlessly pull even the most obscure file-types. In most cases, data-sets are available in a .csv ("comma-separated values", basically universal spreadsheet file type) format. Or, it might be in an Excel spreadsheet specifically, which Stata has very good utilities for handling (see the `help` file; it can do things such as automatically use column headers as variables names or only import certain ranges or sheets of the total file).
- If your data are in "wide" format, i.e. you have some time-series data, and the year-level information is stored as part of the column name rather than as a "multi-index" or hierarchical data, you should reshape with Stata. More concretely, what I mean is if you have data structured like these…

Country	*GDP*1980	*GDP*1990
Country A	y_{1980a}	y_{1990a}
Country B	y_{1980b}	y_1990b

  …you want to get them structured like these…

Country	*Year*	*GDP*
Country A	1980	y_{1980a}
Country A	1990	y_{1990a}
Country B	1980	y_{1980b}
Country B	1990	y_{1990b}

  You should generally do so by `reshape`-ing your data using Stata's utility of the same name. I'll spare you the details in this text, but this is a very common problem. There is a way to do this in Excel, and it is possible that there is some utility in giving you a tactile sense for how that should look, but I would generally not rely on doing this by hand. For details, you can see my SOC365 lectures (available here).
- Similarly, any merges or appends that you need to do should be done in Stata. A merge conceptually means that you are "adding columns" to the data: you're doing something like matching people who are, for whatever reason, split across two different files (as in the NHANES data-sets as they

[1]The better you get at computing and research, the more you will come to hate the GUI; a two-day project I recently executed involving learning a system (Vim) to literally not need a mouse for any computing task. This is a) so much faster once you get good at typing and b) so much more well-documented; keystrokes leave simple forensic records, and mouseclicks

come to us raw; you all have only seen my merge files in this class), which is called a $1:1$ merge; or, the other common situation, you might be matching individuals with group-level information (e.g., you might know what state someone lives in but the data-set may not include that state's unemployment information, wich perhaps you have stored in a separate file—you can, however, simply merge the two files).

5 Writing it all up

In this section, I won't provide tips so much as a guideline for how to write all of this up. In Stata, you have a few options for exporting your regression results. I will say in advance that Stata is generally worse at this than other software, which is one reason to be "multilingual" in terms of programming languages. The paper production pipeline is generally easier with software which not only incorporates some basic programming tools but also makes them natural to learn and easy to use. Stata generally does not do a great job of this (more on that later). Until recently, the state of the art here was user-written commands (generally not a good thing for proprietary software); I'll try to show you what I know of the "state of the art" now, but one reason that I moved on from Stata is that this part was always such a pain.

Once you have your results up in Stata, you generally want to export them. The best way to do this is probably to use Stata's `etable` syntax; an alternative is discussed here and compared to the `etable` routine. `etable` builds off of the `estimates store` approach that we've seen before. Here is a simple example of the routine. Note that you don't need to run a loop to successively incorporate predictors, as I did. That is just one way to speed it up.

```
cd ~/desktop/code/soc361fa24/soc361fa24week13
sysuse auto, clear
local predictors mpg weight trunk
local model = ""
local i = 0
foreach var of varlist `predictors' {
    local i = `i' + 1
    di `i'
    local model = "`model'" + " " + "`var'"
    reg price `model'
    estimates store model`i'
}
est table model*, b(%9.2f) se(%9.2f) p(%9.2f)
local fo = "nformat(%9.2f)"
etable, estimates(model*) ///
    cstat(_r_b, `fo') cstat(_r_se, `fo') cstat(_r_p, `fo') ///
    title("Table 1. Models for price") ///
    export(./regresults.docx, replace)
```

The main thing to note here is that we just add the command `etable` and slightly adjust how we call statistics. The stuff with `b` is for β and means our estimates of those parameters; `se` is standard error and `p` is a $p-$value. Stata is somewhat annoying about making you specify each statistic that you want reported and its formatting, so to save space I just made a local with the same standard way to format numbers better called `fo` and invoked it in each instanace necessary. Note that the result is a separate `.docx` file; you can probably find a way to get this to go directly into a paper you're writing, but it is messier than with other software.

And here is how to take some results from, say, `marginsplot` and turn them into a graph that you can easily tweak without needing the GUI editor.

```
reg price i.foreign##c.mpg
qui sum mpg
local min = round(r(min), 1)
local max = round(r(max), 1)
local incr = round(r(sd)/4, 1)
margins, at(mpg=(`min'(`incr')`max')) over(foreign)
marginsplot, ///
    title("Different effects of MPG on price by import status") ///
    ytitle("Predicted price")
graph export ./figs/fig1.png, replace
```

Some parts of that are a little technical, but I've explained them all before. Note that I am omitting certain types of models you *should* include for full credit, e.g. non-linear specifications and so on. This is just a small thing that I threw together to demonstrate effective Stata code.

6 How it might look all put together

In the Github directory where you obtained this file is my own paper for this class from n years ago. It was for a different professor, although the same book was used; that class was perhaps slightly more technically-oriented. There are plenty of things I'd change about that paper, but for what it's worth, it was an A paper and the only critical comments largely had to do. I would probably find a lot of mistakes if I went back and gave it a close read, but this passed muster for the course, for what it's worth. I will say, from giving a quick scan, that I was better than I remembered at expressing thoughts clearly and general organization; I think that this paper is a reasonable model to follow, as is my do-file (again, there are many aspects that I would now change and make more concise). You can give the paper a read if you like. If there is interest, I will post a draft of my current work as well that I presented at the American Sociological Association annual meeting in Montreal this summer, but no need to hold yourself to that standard.

7 Why be multilingual in programming? What does this workflow lack?

This workflow I've presented here is partial and includes some data management tips that might only be mildly relevant now but will doubtless be more relevant to many of you later. I want to highlight just a few key issues here that you might want to consider thinking about—if you liked this course at all and have some free time to learn some new stuff that is generally useful to you but kind of a long-term payoff, here are some general comments on Stata vs. competing software, with special emphasis on Python (what I use). Most of this is independent of what comes above (i.e., you can level up your workflow with what came above without necessarily needing to consider what follows).

- Stata is generally kind of challenging to use for pure programming tasks, though not impossible. For example, the fact that its locals do not persist if you run code out of the do-file is irritating; in some ways of scripting, e.g. writing code for R or Python in Jupyter Notebook, the "state" persists by default, which can cause problems in its own way, but it makes it much easier to toggle code to see how you might want to define a local within a loop. Other aspects of the programming process are easier, too: there are no weird quotation marks necessary to invoke a local, and you do not need to write `local` in most other languages; you just write, e.g. `predictors = mpg weight turn` (Cf. before).

  One also can generally write more customizable code; for example, in the process of writing a loop to successively incorporate regression predictors above, I realized that Stata does not have a natural way to invoke the position of an item in a local. It is also generally awkward to, say, store a list of numbers that you want to use as x-ticks for a graph or inputs for a curve that you want to plot points against.

  Perhaps the most general way to say this is that Stata does not do super well with data structures that aren't columns of actual observations; however, for many advanced purposes *or* simply easy customization of figures, you may want easy manipulation of lists of numbers or strings, which, e.g., Python is very good at.

- Relatedly but separate, it is much easier to invoke returned results in Python or R. Remember how often we wanted to do that? Virtually every lab it was useful for some kind of simplification of a programming task. In Python, e.g., one simply writes something such as `gss["educ"].mean()` to obtain the mean of education, *whether or not we have run a command like* `summarize` *(well, Python equivalent) to get it into memory.* In my view, that's a pretty huge advantage. It also is generally much easier to calculate custom summary statistics easily (e.g. a husband-wife income ratio or the coefficient of variation of income in a given year) for that reason, and it is *dead simple* to produce custom tables full of whatever

results you want that you can export, unlike in Stata, which has only belatedly introduced that with the fairly intimidating `collect` suite of commands—a basic feature like "can I put together a table of custom summary statistics for my data" should not require so much work.

- In conjunction with Markdown or LaTeX, it is much easier in most other languages to write a paper that automatically updates tables as you subtly tweak the analysis (and same for figures); there is no need to manually copy/paste all five figures and all three graphs every time they change subtly (trust me, this can start to seem extremely tedious when you're making edits). Plus, Markdown allows you to easily incorporate mathematical notation and many other kinds of specialty text (e.g. non-English diacritics) that can be useful. For actually writing papers, this is extremely useful; you no longer have to worry about endless Word files and how they all relate. There are no real formatting issues because Markdown essentially is the state of the art.
- Stata is also not free and is quite expensive at that.
- This isn't even getting into the question of actual statistical analyses; Stata has the edge on many specialty, obscure "textbook" flavors of regression, but Python is better at machine learning techniques that are generally more interesting to many (such as regression trees or cluster analysis, still tricky in Stata). Python is also the clear winner when it comes to general data management techniques; for example, if you wanted to scrape some websites or even just the text of, say, Congressional hearings, Stata can help you, but the old advice was to reach for supplementary software such as NVivo. Python has excellent tools for this sort of thing because it is also very commercially useful (e.g. the `BeautifulSoup` library).